Interspeech 2024
1-5 September 2024, Kos, Greece

Text-aware Speech Separation for Multi-talker Keyword Spotting

Haoyu Li[1], *Baochen Yang*[1], *Yu Xi*[1], *Linfeng Yu*[1], *Tian Tan*[1], *Hao Li*[2], †*Kai Yu*[1]

[1]MoE Key Lab of Artificial Intelligence, AI Institute, X-LANCE Lab, Shanghai Jiao Tong University, China
[2]AISpeech Ltd, China

{haoyu.li,baochen1202,yuxi.cs,ylf2017,tantian,kai.yu}@sjtu.edu.cn, hao.li@aispeech.com

Abstract

For noisy environments, ensuring the robustness of keyword spotting (KWS) systems is essential. While much research has focused on noisy KWS, less attention has been paid to multi-talker mixed speech scenarios. Unlike the usual cocktail party problem where multi-talker speech is separated using speaker clues, the key challenge here is to extract the target speech for KWS based on text clues. To address it, this paper proposes a novel *Text-aware Permutation Determinization Training* method for multi-talker KWS with a clue-based *Speech Separation* front-end (TPDT-SS). Our research highlights the critical role of SS front-ends and shows that incorporating keyword-specific clues into these models can greatly enhance the effectiveness. TPDT-SS shows remarkable success in addressing permutation problems in mixed keyword speech, thereby greatly boosting the performance of the backend. Additionally, fine-tuning our system on unseen mixed speech results in further performance improvement.

Index Terms: multi-talker keyword spotting, text-aware speech separation, robustness

1. Introduction

Keyword Spotting (KWS), also known as Wake Word Detection (WWD), serves as a critical interface for enabling intelligent interaction on a vast array of edge devices. Despite the substantial advancements that have led to impressive performance benchmarks, KWS systems face significant hurdles in complex acoustic environments. These environments are often marred by environmental noise or overlapping interference from multiple speakers, posing substantial challenges to the system's efficacy. Such conditions significantly compromise the reliability of the wake-up functionality, underscoring the need for solutions to improve system robustness. A widely adopted strategy for improving robustness involves integrating a Speech Enhancement (SE) front-end before the KWS module [1, 2, 3, 4]. These works adopt joint training, curriculum training strategies, self-supervised wav2vec [5], etc., to enhance the resilience of KWS systems in noisy scenarios.

However, addressing environmental noise alone is insufficient for robust KWS systems. Interference from other speakers, i.e., overlapping speech, substantially impacts KWS performance more than ambient noise. Overlapping speech contains segments similar to the wake-up word and is challenging to eliminate without specialized design, which increases the risk of false alarms. On the other hand, if the false alarm rate is reduced by raising the threshold, the wake-up rate is greatly affected. This paper focuses on the robust KWS against multi-talker interference. Unlike the usual cocktail party problem where multi-talker speech is separated according to speaker clue, the focus here is to extract the correct speech channel containing the target keyword. To the best of our knowledge, little prior work has explicitly addressed the KWS task-specific multi-talker problem.

In [6], the authors introduce TDSE, a SE front-end designed to simultaneously eliminate ambient and human noise for KWS. Each training sample for TDSE is synthesized with a keyword to ensure targeted noise suppression and enhanced keyword recognition. However, the absence of negative training data makes the model susceptible to overfitting, resulting in a high false alarm rate. As for multi-talker overlapping speech, training a speech separation model using PIT [7, 8] criterion is a highly effective technique. It calculates the loss of all possible permutations of the separated signals and selects the permutation with the minimum loss to optimize the model. PIT is initially proposed for signal-level separation and subsequently applied to multi-talker automatic speech recognition (ASR) [9, 10, 11, 12, 13]. While PIT has demonstrated superior separation performance, directly applying it for multi-talker KWS is unreasonable. The main reason is PIT regards all output channels equally, but for the KWS task, we only care about the keyword output channel. Recently, keyword clue-based frameworks [14, 15, 16, 17, 18, 19] are widely explored under different sub-tasks of KWS, like open-vocabulary KWS or noise robust KWS. The keyword clue information can effectively bias the output of models and boost performance. If we feed separated speech from all channels to the KWS model instead of the biasing channel, the processed data of the backend will double (assuming the number of output channels is 2). This increases not only the risk of more false alarms but also the computational burden of the backend, which is unsuitable for an on-device system. Thus, we propose the TPDT-SS front-end, a text-aware speech separation model tailored for noisy multi-talker KWS. Our contributions are summarized as follows.

- We demonstrate the importance of a speech separation front-end for multi-talker KWS and propose a novel TPDT-SS approach to enhance SS by incorporating keyword clues and permutation determinization training (PDT). Furthermore, we explore and identify potential methods for integrating keyword information into SS. Codes are open-sourced here[1].
- We demonstrate that channel permutation, which significantly impacts the KWS performance, is effectively mitigated by PDT. We observe a significant improvement in KWS tasks on audios processed by TPDT-SS.
- We fine-tune the KWS model using unseen multi-talker speech processed by TPDT-SS, further improving KWS per-

†Kai Yu is the corresponding author.

[1]https://github.com/GnafiY/TPDT-SS-KWS

10.21437/Interspeech.2024-789

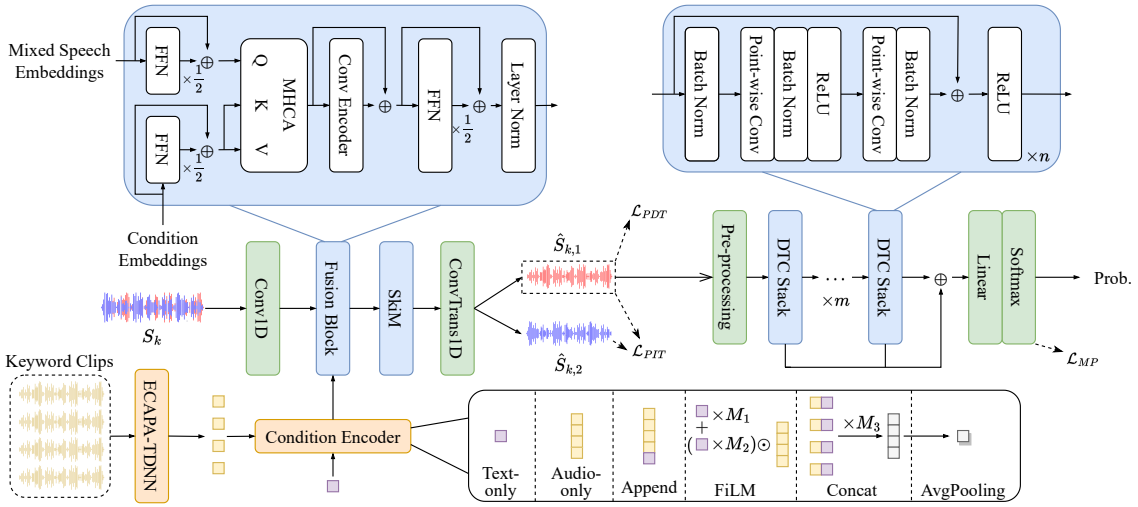

Figure 1: *The overview of the whole system: TPDT-SS front-end(TSkiM) and MDTC KWS backend.*

formance. This demonstrates the potential of fine-tuning KWS models through the real mixed data from our proposed front-end.

2. Text-aware multi-talker KWS

A text-aware multi-speaker system includes a TPDT-SS front-end and a KWS module. Using the keyword as a clue, the TPDT-SS front-end isolates the clear keyword speech into the preset channel while distributing the remaining speech to another channel under the assumption of two speakers. If the multi-speaker speech lacks the keyword, traditional PIT-based speech separation techniques are applied. The KWS module then processes the clear speech from the preset channel to determine the presence of the keyword. Further information is available in Figure 1.

2.1. The overview of TPDT-SS

From an architectural standpoint, the proposed system's primary distinction lies in adding a keyword condition module, which extracts the desired keyword speech from mixed speech. This module's design draws inspiration from incorporating target speaker embeddings into mixed speech for target speaker extraction (TSE) tasks, as outlined in [20]. Specifically, we introduce a keyword condition encoder designed to encode various types of keyword modality information, such as text-based wake words or a specific number of wake word audio clips. This keyword information is then integrated with mixed speech embeddings through an attention-based mechanism, which is essential for embedding the keyword clue into the SS process. Subsequently, the front-end module proceeds as a standard SS model, segregating the audio of different speakers into separate channels. For processing mixed speech, we employ SkiM [21], a model recognized for its lightweight design, low latency, and real-time SS capabilities, making it well-suited for KWS.

2.2. Keyword information extraction

We integrate diverse numbers, modalities, and merging functions within the front-end to maximize the efficacy of the keyword condition module. We create an embedding vector to represent textual keyword clues, and for audio keyword clues, we employ the ECAPA-TDNN module [22]. Both the textual and audio clue vectors are trained from scratch alongside the other components of the TPDT-SS system. In the subsequent sections, which focus on experiments and results, our front-end model, TPDT-SS, integrates the SkiM module, referred to as Text-aware SkiM (TSkiM). Our experiments leverage different types of clues, including text-only, audio-only, as well as combinations of text and audio clues. More details are provided in the bottom portion of Figure 1.

2.3. MDTC KWS module

We utilize the Multi-scale Depthwise Temporal Convolution (MDTC)[23] KWS model, designed to extract multi-scale features through varied receptive fields, as illustrated in the top right section of Figure 1. To train the KWS backend, we employ a refined max-pooling loss [24], a practical training criterion adapted from its initial formulation [25]. Further specifics about this loss are detailed in [24].

2.4. Training and inference

The backbone of TPDT-SS can be divided into three parts: the text and/or audio condition encoder, attention-based conformer [26] fusion block, and SkiM [21] separator as mentioned in Section 2.1. We adopt the time-domain SI-SNR loss[27] to optimize the model. Specifically, for overlapping speech S_k, which is mixed by reference $S_{k,1}$ and $S_{k,2}$, we denote separated outputs by $\hat{S}_{k,1}$ and $\hat{S}_{k,2}$, where $\hat{S}_{k,1}$ always contains keyword if S_k is a keyword-mixed speech and $\hat{S}_{k,2}$ is keyword-unrelated.

Whether the training speech contains the keyword or not, the front-end model always applies the PIT criterion just like general SS models:

$$\mathcal{L}_{PIT}(S_k) = \min_{\sigma}\{-\text{SI-SNR}(\hat{S}_{k,1}, S_{k,\sigma(1)}) - \text{SI-SNR}(\hat{S}_{k,2}, S_{k,\sigma(2)})\}, \quad (1)$$

where σ denotes one of the two (2!) possible permutations. Then, the separation model is optimized by the permutation with minimum loss. For training data that contains the keyword, the model is additionally guided by $\mathcal{L}_{PDT}$ to separate the keyword speech to the preset channel and output residual parts to another channel. The scale-invariant signal-to-noise ratio (SI-SNR) loss of PDT can be formulated as follows:

$$\mathcal{L}_{PDT}(S_k) = -\text{SI-SNR}(\hat{S}_{k,1}, S_{k,1}) - \text{SI-SNR}(\hat{S}_{k,2}, S_{k,2}), \quad (2)$$

where the permutation of the output is preset. The total loss $\mathcal{L}$ of S_k is the summation of the two kinds of training data:

$$\mathcal{L}(S_k) = \mathcal{L}_{PIT}(S_k) + y_k\mathcal{L}_{PDT}(S_k), \quad (3)$$

where y_k equals 1 if S_k contains keyword, otherwise 0.

3. Experimental setups

This section outlines the construction of datasets and the setup for both the TPDT-SS front-end and the MDTC KWS backend. Subsequently, we detail the KWS performance metrics employed for evaluating the multi-speaker KWS task. The baseline SkiM is available as an open-source tool in ESPnet-SE [28].

3.1. Dataset

There are three parts of data to evaluate the performance: mixed multi-talker general ASR data, environmental noise, and mixed keyword data. The following are the details of the used datasets.

- **Libri2Mix (L2M)**. Libri2Mix [29] is a speech separation corpus with ambient noise. It is derived from LibriSpeech [30] clean subset and WHAM! [31] noise. Each sample in Libri2Mix is synthesized by two different audios from LibriSpeech and a piece of ambient noise from WHAM!. We follow the official scripts to prepare general multi-talker ASR data.
- **Snips2Mix (S2M)**. Snips2mix is a self-constructed multi-talker KWS dataset containing the keyword "Hey Snips" in each sample. Hey Snips [32] is an open-source KWS dataset that specifically uses "Hey Snips" as the keyword. Each sample in Snips2Mix is constructed by mixing one item from each of three datasets: LibriSpeech clean part, Hey Snips positive part, and WHAM! noise. We implement our simulated scripts based on the Libri2Mix data preparation scripts. They are open-sourced along with the codebase.
- **Snips2Mix-2000 (S2M-2000)**. This is an additional 2000 training pieces of training data simulated with the same scripts as Snips2Mix, designed to mimic the unseen data generated by users. This data has no clean audio reference and is used to fine-tune the backend to further validate the model's generalization and potential.
- **Hey Snips (Snips)**. We have mentioned the Hey Snips dataset in Snips2Mix before. We also use the original clean Hey Snips dataset to train our backend MDTC KWS model. The details of the dataset can be found in [32]. Unless otherwise noted, our base KWS models are trained on the clean Hey Snips dataset.

To create a multi-speaker dataset for SS, we combine Libri2Mix and Snips2Mix. For the evaluation of KWS models, a test dataset is constructed, comprising 4.2 hours of negative samples and 1.9 hours of positive samples. Detailed information about this dataset is provided in Table 1.

3.2. TPDT-SS front-end configurations

SkiM open-sourced in ESPnet-SE toolkit [28] serves as the baseline for SS. The TPDT-SS front-end is developed using the ESPnet toolkit [33]. Each SS model undergoes training for up to 100 epochs, employing a batch size of 64 and chunk iterators set to 24,000. The training process utilizes the Adam optimizer [34] with a learning rate of 1e-3. The configurations for all hyper-parameters used during training, and all training details, along with the codes, are published in the codebase.

The condition list generated by randomly selecting audio segments from Hey Snips is fixed during training and inference inspired by the prior work [19]. We conduct experiments with 0, 10, 20, 50, and 100 keyword clips as audio conditions to find the optimal condition option for TPDT-SS. Each TSkiM model in Section 4 is trained with text conditions. For example, TSkiM represents the text-only condition, and TSkiM-10 represents the combination of text and 10 fixed audio segments. By the way, the ECAPA-TDNN module is discarded after extracting speech conditions for inference, which saves a lot of memory and computational resources.

Table 1: *The number of utterances of datasets. "-" means none and "/" represents positive/negative.*

Dataset	Model	Train	Dev	Test
L2M	TPDT-SS	13900	1500	3000 (4.2hrs)
S2M		5000	1500	3000 (1.9hrs)
Snips	MDTC	5799/44859	2484/20179	2529/20543
S2M-2000		2000	-	-

3.3. MDTC KWS backend configurations

We leverage the MDTC KWS backend implemented in WeKws[24] and use the default configuration. The default backend KWS model is trained on the Hey Snips dataset. We evaluate the backend performance through the metrics: recall (Rec) and false alarms per hour (FA/h). Recall comparison is conducted under 0.5 FA/h in the following results section. To better compare the KWS performance for different SS models, we employ two selection methods to merge backend results:

- **Max Selection (MAX)**. We apply the keyword spotting module to both output channels, and the maximum score will be selected as the final output. The cost is that the backend inference computation grows with the number of channels.
- **Channel-1 Selection (CH1)**. Since the TPDT-SS model has acquired the capability to determine the permutation, there is no need for extra channel selection. Thus, we will only apply the KWS module to the output of the first channel (preset as the keyword channel).

4. Results and analysis

4.1. The importance of SS front-end for multi-talker KWS

We first explore the feasibility of training the KWS backend on mixed speech. We can conclude from Table 2 that the KWS module trained on clean data performs exceptionally poorly on mixed speech, which indicates that the KWS model trained on clean data is not robust against noisy environment. Then, we train the KWS model on Hey Snips, Snips2Mix, and Libri2Mix datasets to test whether the KWS model converges on mixed speech data. Results notify us that the KWS module trained on mixed data also performs poorly on mixed data (82.83% vs 97.43%). We also adopt a naive speech enhancement (SE) version SkiM (only one output channel) trained by time-domain SI-SNR loss [27] like the work [19]. Although the SE front-end works well in [19], adding speech noise in training data does not perform well (83.63% vs 82.83%). Besides, additional mixed data does not help; instead, it degrades the performance further by adding more noise (76.10% vs 82.83%). The results indicate the task is challenging to tackle with only the KWS backend or adding a SE front-end and requires an SS front-end to process multi-talker speech.

Table 2: *Results of KWS models trained on clean/mixed speech.*

Training Data	Test Data	Front-end	Recall(%)
Snips	Snips	-	97.43
	L2M + S2M	-	19.23
Snips + L2M + S2M	L2M + S2M	-	82.83
		SE	83.63
Snips + L2M + S2M + S2M-2000		-	76.10

Table 3: *Results of TSkiMs on different audio-text information fusion methods. The number of audio clips on the front-end models is all set to 10.*

Model	SS (S2M)		SS (L2M)		KWS
	STOI	SI-SNR	STOI	SI-SNR	CH1(%)
Text-only	85.96	9.71	84.52	9.52	**92.80**
Audio-only	86.02	9.87	84.73	**9.65**	89.63
Append	86.15	9.95	84.58	9.55	91.60
FiLM [35]	84.23	9.30	84.16	9.31	91.30
Concat	**86.29**	**9.95**	**84.86**	9.63	91.73
Concat + AvgPooling	86.22	9.90	84.76	9.53	90.60

4.2. Keyword clue fusion methods

As shown in Figure 1, the fusion block takes audio and/or text clues of the keyword as the condition. Appropriate methods to fuse keyword-related information from the text and audio modals are worth exploring. We primarily examine four fusion methods, including text-only and audio-only.

We analyze the results in Table 3 from two aspects. First, the text-only system gets the best KWS results (92.80%). We believe that this is because text information is the most relevant and purest condition for the KWS task compared with audios, which contains much extra acoustic information, like record environments, speakers, etc. Most models with audio conditions achieve better separation results than the text-only system. This is because the richer information contained in audios is helpful for separation. Though the model with text-only conditions achieves the best KWS recall, we still think the front-end performance dramatically impacts improving KWS performance. Therefore, we conduct further experiments on the model with the concatenation (concat) fusion method.

4.3. Performance of KWS with TPDT-SS

In this section, we evaluate the proposed TSkiM, i.e., KWS with the proposed TPDT-SS, on datasets Libri2Mix (L2M) and Snips2Mix (S2M). In the upper part of Table 4, we evaluate the importance of the loss derived in Equation (3). From the lines of model (A) and model (C), we see that although the results of the front-end trained by PIT-only are good, the KWS results are not excellent enough compared with model (D)-(H). Besides, it's worth noting that the recall of channel one is nearly half of the max selection method, which indicates the nature of PIT: regarding the two output channels equally. That's why we need PDT. We also train the model (B) with only PDT and without PIT. The output channel for keywords is fixed, as the results are nearly the same between MAX and CH1. However, the front-end performance degrades dramatically, especially on Libri2Mix (SI-SNR=-0.47). Therefore, PIT is also indispensable in improving the results of general data. The model (D), which is trained with PIT and PDT, has relatively acceptable front-end and backend performance. In the bottom part, we further investigate the optimal number of audio clues and find that around 50 is the best. Another reasonable finding is that backend performance is almost positively correlated with front-end performance: the higher the separation metrics, the better the performance of the KWS system.

4.4. Fine-tuning on seen and unseen data

Since our default KWS model is trained on the clean dataset and inference with cleaned data, a mismatch exists, and the model could be further optimized. This section investigates how the backend model performs after fine-tuning on the separated data. To generate the fine-tuning dataset for the KWS model, we run inference on the corresponding SS model ((A), (G) front-ends in Table 4) with datasets S2M (seen) and S2M-2000 (unseen). Fine-tuned KWS models are all initialized from the clean checkpoint ((A), (G) backends in Table 4). We also add L2M data processed by SS models to the fine-tuning dataset to avoid overfitting the keyword.

Table 4: *Performance of SS models trained on both **S2M** and **L2M**. TSkiM-n means the fusion block of TSkiM encodes n keyword clips as audio conditions. All front-end models use the same KWS backend trained on clean Hey Snips.*

ID	Model	$\mathcal{L}_{PIT}$	$\mathcal{L}_{PDT}$	SS (S2M)		SS (L2M)		KWS	
				STOI	SI-SNR	STOI	SI-SNR	MAX(%)	CH1(%)
(A)	SkiM[21]	✓	N/A	**86.04**	**9.93**	**84.93**	**9.74**	82.80	41.77
(B)	TSkiM	✘	✓	80.38	5.82	66.23	-0.47	88.53	88.47
(C)	TSkiM	✓	✘	85.91	9.78	84.62	9.59	79.70	40.13
(D)	TSkiM	✓	✓	85.96	9.71	84.52	9.52	**92.20**	**92.80**
(E)	TSkiM-10	✓	✓	86.29	9.95	**84.86**	9.63	92.07	91.73
(F)	TSkiM-20	✓	✓	86.21	9.76	84.62	9.52	90.83	92.70
(G)	TSkiM-50	✓	✓	**86.33**	**10.00**	84.76	**9.66**	**94.90**	**95.27**
(H)	TSkiM-100	✓	✓	86.15	9.91	84.76	9.59	90.23	90.87

Table 5: *Recall(%) of fine-tuning KWS models by separated seen and unseen data.*

Model	No FT(%)	FT on seen data(%)	FT on unseen data(%)
SkiM	82.80	93.27	80.73 (CH1) or 93.67 (MAX)
TSkiM-50	**95.27**	**96.77**	**97.57**

From Table 5, we can see the consistent improvement of the models when fine-tuning on the seen training data inferred by the corresponding front-end models. However, with the addition of unseen data, models behave differently. The new training data generated from TSkiM are mostly keyword clips as the majority of keyword audios are separated into the preset channel. Thus, the system has been further significantly improved. Conversely, SkiM does not have this advantage, so we have to randomly select a channel (CH1) or utilize the KWS model to score each separated segment and choose the higher one manually (MAX). The two selection methods do not perform well. MAX selection has a slight gain but needs the backend system to score each segment, which adds the computational burden. The CH1 selection method is not a reasonable way to pick the data, so the fine-tuning result with the online data is even worse than the clean result. The results in Table 5 indicate the potential and generalization of our proposed model.

Limitations. The front-end SS model increases the computational resource and memory consumption, which is a burden for the device system. In the future, we aim to conduct joint training between front-end and backend models, making TPDT-SS better fit the KWS task.

5. Conclusion

In this paper, we propose TPDT-SS, a novel text-aware speech separation model for multi-talker KWS. First, we evaluate the importance of the SS module. Then, we adopt multiple ways to construct keyword clues to bias the output of the front-end SS. The proposed PDT is effective at fixing the output channel of keyword speech. Compared with the original SS baseline trained with only PIT, the proposed model not only saves the computational resource as it only infers one channel data for the KWS backend but also improves the SS and KWS performance significantly. In addition, our proposed model can further perform better by fine-tuning with actual online multi-talker KWS data, while cannot be done by PIT-based models, indicating the superior application prospect of our methods.

6. Acknowledgements

This work was supported by Shanghai Municipal Science and Technology Major Project (2021SHZDZX0102), and Key Research and Development Program of Jiangsu Province, China (Grant No. BE2022059).

7. References

[1] D. Bonet, G. Cámbara, F. López, P. Gómez, C. Segura, J. Luque, and M. Farrús, "Speech Enhancement for Wake-Up-Word detection in Voice Assistants," in *Proc. ISCA Interspeech*, 2021, pp. 41–45.

[2] C. Yang, Y. M. Saidutta, R. S. Srinivasa, C.-H. Lee, Y. Shen, and H. Jin, "Robust Keyword Spotting for Noisy Environments by Leveraging Speech Enhancement and Speech Presence Probability," in *Proc. ISCA Interspeech*, 2023, pp. 1638–1642.

[3] D. Ng, Y. Chen, B. Tian, Q. Fu, and E. S. Chng, "Convmixer: Feature interactive convolution with curriculum learning for small footprint and noisy far-field keyword spotting," in *Proc. IEEE ICASSP*, 2022, pp. 3603–3607.

[4] M. N. Ali, D. Falavigna, and A. Brutti, "Enhancing embeddings for speech classification in noisy conditions," in *Proc. ISCA Interspeech*, 2022, pp. 2933–2937.

[5] S. Schneider, A. Baevski, R. Collobert, and M. Auli, "wav2vec: Unsupervised Pre-Training for Speech Recognition," in *Proc. ISCA Interspeech*, 2019, pp. 3465–3469.

[6] M. Yu, X. Ji, Y. Gao, L. Chen, J. Chen, J. Zheng, D. Su, and D. Yu, "Text-Dependent Speech Enhancement for Small-Footprint Robust Keyword Detection," in *Proc. ISCA Interspeech*, 2018, pp. 2613–2617.

[7] D. Yu, M. Kolbæk, Z.-H. Tan, and J. Jensen, "Permutation invariant training of deep models for speaker-independent multi-talker speech separation," in *Proc. IEEE ICASSP*, 2017, pp. 241–245.

[8] M. Kolbaek, D. Yu, Z. Tan, and J. H. Jensen, "Multi-talker speech separation with utterance-level permutation invariant training of deep recurrent neural networks," *IEEE/ACM Trans. ASLP.*, pp. 1901–1913, 2017.

[9] D. Yu, X. Chang, and Y. Qian, "Recognizing Multi-Talker Speech with Permutation Invariant Training," in *Proc. ISCA Interspeech*, 2017, pp. 2456–2460.

[10] Y. Qian, X. Chang, and D. Yu, "Single-channel multi-talker speech recognition with permutation invariant training," *Speech Communication*, pp. 1–11, 2018.

[11] X. Chang, W. Zhang, Y. Qian, J. Le Roux, and S. Watanabe, "Mimo-speech: End-to-end multi-channel multi-speaker speech recognition," in *Proc. IEEE ASRU*, 2019, pp. 237–244.

[12] X. Chang, W. Zhang, Y. Qian, J. L. Roux, and S. Watanabe, "End-to-end multi-speaker speech recognition with transformer," in *Proc. IEEE ICASSP*, 2020, pp. 6134–6138.

[13] W. Zhang, X. Chang, C. Boeddeker, T. Nakatani, S. Watanabe, and Y. Qian, "End-to-end dereverberation, beamforming, and speech recognition in a cocktail party," *IEEE/ACM Trans. ASLP.*, pp. 3173–3188, 2022.

[14] Y. Xi, T. Tan, W. Zhang, B. Yang, and K. Yu, "Text adaptive detection for customizable keyword spotting," in *Proc. IEEE ICASSP*, 2022, pp. 6652–6656.

[15] A. Zhang, P. Zhou, K. Huang, Y. Zou, M. Liu, and L. Xie, "U2-KWS: unified two-pass open-vocabulary keyword spotting with keyword bias," in *Proc. IEEE ASRU*, 2023, pp. 1–8.

[16] A. Navon, A. Shamsian, N. Glazer, G. Hetz, and J. Keshet, "Open-vocabulary keyword-spotting with adaptive instance normalization," in *Proc. IEEE ICASSP*, 2024, pp. 11 656–11 660.

[17] Y. Xi, B. Yang, H. Li, J. Guo, and K. Yu, "Contrastive learning with audio discrimination for customizable keyword spotting in continuous speech," in *Proc. IEEE ICASSP*, 2024, pp. 11 666–11 670.

[18] H.-K. Shin, H. Han, D. Kim, S.-W. Chung, and H.-G. Kang, "Learning audio-text agreement for open-vocabulary keyword spotting," in *Proc. Interspeech*, 2022.

[19] S. Lv, X. Wang, S. Sun, L. Ma, and L. Xie, "DCCRN-KWS: An Audio Bias Based Model for Noise Robust Small-Footprint Keyword Spotting," in *Proc. ISCA Interspeech*, 2023, pp. 929–933.

[20] K. Zmolíková, M. Delcroix, T. Ochiai, K. Kinoshita, J. Cernocký, and D. Yu, "Neural target speech extraction: An overview," *IEEE Signal Process. Mag.*, pp. 8–29, 2023.

[21] C. Li, L. Yang, W. Wang, and Y. Qian, "Skim: Skipping memory lstm for low-latency real-time continuous speech separation," in *Proc. IEEE ICASSP*, 2022, pp. 681–685.

[22] B. Desplanques, J. Thienpondt, and K. Demuynck, "ECAPA-TDNN: emphasized channel attention, propagation and aggregation in TDNN based speaker verification," in *Proc. ISCA Interspeech*, 2020, pp. 3830–3834.

[23] J. Hou, L. Xie, and S. Zhang, "Two-stage streaming keyword detection and localization with multi-scale depthwise temporal convolution," *Neural Networks*, pp. 28–42, 2022.

[24] J. Wang, M. Xu, J. Hou, B. Zhang, X. Zhang, L. Xie, and F. Pan, "Wekws: A production first small-footprint end-to-end keyword spotting toolkit," in *Proc. IEEE ICASSP*, 2023, pp. 1–5.

[25] M. Sun, A. Raju, G. Tucker, S. Panchapagesan, G. Fu, A. Mandal, S. Matsoukas, N. Strom, and S. Vitaladevuni, "Max-pooling loss training of long short-term memory networks for small-footprint keyword spotting," in *Proc. IEEE SLT*, 2016, pp. 474–480.

[26] A. Gulati, J. Qin, C. Chiu, N. Parmar, Y. Zhang, J. Yu, W. Han, S. Wang, Z. Zhang, Y. Wu, and R. Pang, "Conformer: Convolution-augmented transformer for speech recognition," in *Proc. ISCA Interspeech*, 2020, pp. 5036–5040.

[27] Y. Luo and N. Mesgarani, "Conv-tasnet: Surpassing ideal time-frequency magnitude masking for speech separation," *IEEE/ACM Trans. ASLP.*, pp. 1256–1266, 2019.

[28] C. Li, J. Shi, W. Zhang, A. S. Subramanian, X. Chang, N. Kamo, M. Hira, T. Hayashi, C. Böddeker, Z. Chen, and S. Watanabe, "Espnet-se: End-to-end speech enhancement and separation toolkit designed for ASR integration," in *Proc. IEEE SLT*, 2021, pp. 785–792.

[29] J. Cosentino, M. Pariente, S. Cornell, A. Deleforge, and E. Vincent, "Librimix: An open-source dataset for generalizable speech separation," *arXiv preprint arXiv:2005.11262*, 2020.

[30] V. Panayotov, G. Chen, D. Povey, and S. Khudanpur, "Librispeech: An asr corpus based on public domain audio books," in *Proc. IEEE ICASSP*, 2015, pp. 5206–5210.

[31] G. Wichern, J. Antognini, M. Flynn, L. R. Zhu, E. McQuinn, D. Crow, E. Manilow, and J. L. Roux, "Wham!: Extending speech separation to noisy environments," in *Proc. ISCA Interspeech*, 2019, pp. 1368–1372.

[32] A. Coucke, M. Chlieh, T. Gisselbrecht, D. Leroy, M. Poumeyrol, and T. Lavril, "Efficient keyword spotting using dilated convolutions and gating," in *Proc. IEEE ICASSP*, 2019, pp. 6351–6355.

[33] S. Watanabe, T. Hori, S. Karita, T. Hayashi, J. Nishitoba, Y. Unno, N. E. Y. Soplin, J. Heymann, M. Wiesner, N. Chen, A. Renduchintala, and T. Ochiai, "Espnet: End-to-end speech processing toolkit," in *Proc. ISCA Interspeech*, 2018, pp. 2207–2211.

[34] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *International Conference on Learning Representations, ICLR*, 2015.

[35] E. Perez, F. Strub, H. de Vries, V. Dumoulin, and A. C. Courville, "Film: Visual reasoning with a general conditioning layer," in *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*, 2018, pp. 3942–3951.